Spitfire ’91, a WPF Shmup

Victor Gasior and Jaxx ’Lucky’ Woods

Abstract

Spitfire ’91 is a retro inspired Shoot ’em Up game (or shmup) where you control a small airplane fighting drones, tanks, and enemy aircraft. The target audience for this project is fans of retro schups like EDF or UN Squadron. So far we have constructed just the basic plans for the project.

1. Introduction

Spitfire ’91 should be a fun, fast paced, retro inspired Shoot ’em Up game made in WPF / C Sharp. In Spitfire ’91 the player controls a modern military fighter jet and fights enemy jets, drones, and ground vehicles. The game is supposed to emulate an SNES (Super Nintendo Entertainment System) era Shoot ’em Up game with its own art, design, sounds, and mechanics. There will be at least one stage where the players can fight enemies in order to try to get a higher and higher high scores. The target audience for this game will be Shoot ’em Up enthusiasts and fans of retro styled video games in general. The audience should first and foremost enjoy the game, while also tailoring the experience to resemble a retro shmup without making it difficult for fans of modern video games to enjoy. There are not many Shoot ’em Up games produced any more so we are appealing to a niche audience starved of content.

1.1. Background

Shoot em’ Ups rely on a player controller a vehicle that can move around the screen and shoot enemies that are trying to attack the player. The player obtains score by avoiding and defeating enemies. The game ends once they reach the end of the level / stage or loses do to being hit by enemies too many times

The reason we decided to undertake this project is that Shoot ’em Ups are both fun to play, and seem interesting to design.

1.1.1. Important Terminology. Shump - Common abbreviation for Shoot ’em Up High Score - A goal of a Shoot ’em Up game, players try to earn higher scores.

1.2. Impacts

This game should bring enjoyment to the audience and provide a free modern alternative to old retro Shoot ’em Up games that can be hard to play due to old hardware and games not being ported to modern systems.

1.3. Challenges

Handling enemy, projectile, and player collision in WPF. Displaying information and moving backgrounds in WPF. Storing enemy and level data in WPF.

2. Scope

This game should the bare minimum contain one playable level with multiple enemy variations that can spawn. There should be a win and lose state, as well as a connected ’Game Over’ screen on loss. There should be a main menu screen the user accesses first before starting the main game. There should be a high score tracker which displays and saves the highest scores to add replay value. Current Stretch Goals are as follows: multiple levels with different backgrounds, a working, upgrade shop and currency system, game saving, and music

2.1. Requirements

The requires come from how we intend the game to operate, and what we think will be the most fun for the user to experience. The requirements both make it so the game is operational (i.e. game actually works, core elements, etc.), while also trying to make the user experience as great as possible. (i.e. sound and visual effect feedback)

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Play Game	user	High	1
2	Check Highscore	user	Med	2
3	Pause Game	user	Easy	3

TABLE 1. SAMPLE USE CASE TABLE

2.1.1. Functional.

- User needs to be able to control the character to avoid enemies and shoot enemies.
- User needs to be able to earn points by eliminating enemies.
- User needs to be able to view their current score.
- Game needs to spawn enemies for the player to both avoid and shoot.
- User needs to be able to lose, then be able to try again.

2.1.2. Non-Functional.

- There should be visual effects and sound effects to enhance the game experience.
- There should be a scrolling background to give the user the impression of flying

2.2. Use Cases

1.

Use Case Number: 1
Use Case Name: Playing the Game
Description: A user wants to play the game, they will open the game and hit 'play' on the main menu.
1) User opens the application.
2) User hits play.
3) User plays the game until loss.
Termination Outcome: The user's score is saved, highest score is updated if this score is the new highest score, which is displayed on the main menu.

Use Case Number: 2
Use Case Name: Check Highscore
Description: A user wants to check the highscore of the local game.
1) User opens the application.
Termination Outcome: The user looks at the score on the main menu.

Use Case Number: 3
Use Case Name: Pausing the Game
Description: A user wants to pause the game during use
1) User is currently playing the game.
2) User pauses the game using the 'p' key
Termination Outcome: The user keeps the game paused for as long as they want, unpausing if they wish by hitting 'p' again

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

We have already made a basic prototype, but we have yet to develop the minimum viable product to show to the class. The steps we will take to create the project will go as follows: 1. Create a game engine that will control several important functions such as: Movement control, enemy spawning, and collision detection. 2. Create the movement controls and projectile firing. 3. Create the spawning of enemies and allow several different versions of enemies to be selected. This includes making an abstract enemy object and its concrete children. 4. Create the system of collision detection. This also controls the losing of health and gaining of points as certain collisions are detected. 5. Create a Game Over state and display the appropriate state for a game over. 6. Create a Pause Function 7. Create a main menu with High Score Board. 8. Polishing Audio and Visual presentation.

Figure 1. This displays an idea for how the game looks during play

4. Project Structure

The project switches between three main windows, that open and close as appropriate. The Main Menu, is exactly what you would think it is, it has the controls for starting and exiting the game as well as displaying the highest scores. The GameMain is what controls the game and is what the player will spend most of their time in. This window is switches to GameOverWindow upon the player losing the game. The GameOverWindow allows the player to go straight to either the GameMain or MainMenu window, as well as displaying the score the player just achieved.

4.1. UML Outline

2 Here you can see the UML for our program. The meat of the program is in the GameEngine class, that contains several important functions that are called by both the game's main window 'GameMain' and the GameOverWindow. The Bullet class is derived from the bullet template (Template Design Pattern) and provides the fireBullet function the Rectangle it needs to add to the screen. The spawnEnemy function needs an Enemy class, including EnemyBasic and it's two children (Factory Design Pattern). The three windows are designed to be swapped between when certain functions are called.

4.2. Design Patterns Used

We are implementing the Factory Pattern and Template Pattern for the project.

5. Results

Turns out WPF is not really meant for making games, and by not really, we mean not at all. WPF is an absolute nightmare to make games for, causing us to have to find weird workarounds and use things in strange unintended ways. Since neither of us took any graphics classes before hand we were stuck wrangling with the default WPF canvas. Currently we got a 'game' where you can move around the plane and shoot, but not much else. As it currently stands, enemy spawning is currently broken, and without enemy spawning various game functions like health, scoring, and the game over screen have no function.

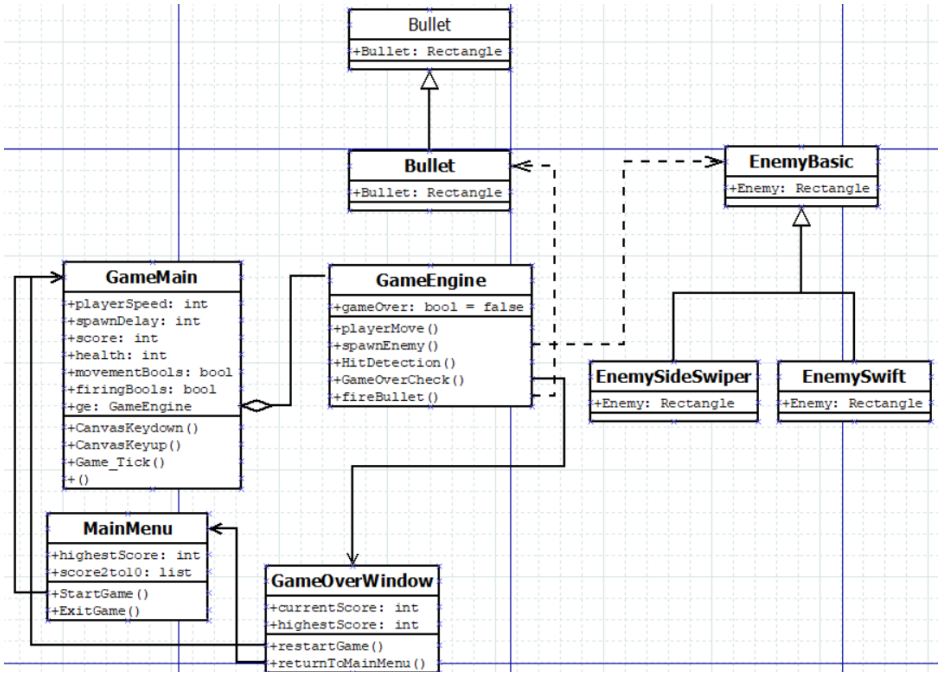

Figure 2. This is the UML

5.1. Future Work

We are about $\frac{3}{5}^{ths}$ done with the project, but we only have so much time left. Highest priority is getting enemies to work, as without enemies key functions of the game like scoring have no purpose or function. We also need to make the Main Menu and Game Over menu, including the saving and displaying of high scores.